Bioinformatics: Tools for Genome Analysis Final Portfolio

Prepared by: Steven Dea

Spring 2019

Table of Contents

Purpose

The purpose of this portfolio is to demonstrate a knowledge and understanding of many current bioinformatic tools and how to apply them in real world situations. Each technology, tool, or database that I have mentioned in this portfolio is described in detail regarding its function as well as has detailed examples of their use cases. All of these detailed examples and use cases have come from either discussion questions, homework assignments, exam questions, or portions of work from my group project. This portfolio's main purpose is to display my functional knowledge of bioinformatics tools for a possible job application.

Gene Prediction

Bacterial

Bacterial DNA is mostly polycistronic. This means that a single mRNA can contain multiple coding regions and thus, multiple genes can be regulated by a single promoter (also known as an operon). To find coding regions in prokaryotic genomic DNA, we can use programs such as ORF Finder, FGENESB, or JHU's own Glimmer software. These different prokaryotic gene prediction softwares are used to determine the coding regions in genomic DNA. Because prokaryotes have no introns, usually the longest ORFs are the actual protein coding regions. The only difficult part can be choosing the correct start codon! Each of these prediction softwares have slightly different uses, so I will now go through their best use cases. ORF Finder shows all of the ORFs in a genomic sequence and can be used to identify predicted genes via selecting the longest sequences that do not overlap. FGENESB automatically annotates prokaryotic genomic data using a Markov chain model to determine coding regions, promoters, operons, translation sites, and termination sites. Glimmer is a unix based software that uses a training data set and a HMM to find long ORFs in a given sequence.

ORF Use Example

Using ORF Finder, report the locations of the three longest coding regions in Bacillus subtilis in addition to determining their reading frames.

The three longest ORFs found in Bacillus subtilis using ORFinder were ORF1, ORF2, and ORF7 of lengths 1497NT, 402NT, and 342NT. Below is a summary image of these three ORFs.

ORF7	+	3	1962	2303	342 | 113
ORF2	+	1	1564	1965	402 | 133
ORF1	**+**	**1**	**46**	**1542**	**1497 | 498**

ORF1 and ORF2 are on the 1 reading frame while ORF7 is on the 3 reading frame.

FGENESB Use Example

Using FGENESB, identify the CDS of a partial sequence of S. helicoides as well as mRNAs predicted that code for those particular CDS.

The first step was to take the partial sequence of S. helicoides and paste it into the FGENESB program. From this, I found 9 CDS listed and 6 mRNAs that were predicted to code for these particular CDS.

N	Tu/Op	Conserved pairs(N/Pv)	S		Start		End	Score
1	1 Op 1	.	+	CDS	635	-	991	117
2	1 Op 2	.	+	CDS	998	-	1141	144
3	2 Tu 1	.	-	CDS	1126	-	1365	73
4	3 Tu 1	.	+	CDS	1334	-	1978	381
5	4 Tu 1	.	+	CDS	2242	-	2463	231
6	5 Op 1	.	+	CDS	2585	-	4003	998
7	5 Op 2	.	+	CDS	4010	-	4678	423
8	5 Op 3	.	+	CDS	4703	-	4768	72
9	6 Tu 1	.	+	CDS	4880	-	5143	169

Glimmer Use Example

Using the command line and the BFX servers at Johns Hopkins University, analyze a partial CDS sequence from Spiroplasma helicoides strain TABS-2.

Genomic data from the Spiroplasma helicoides strain TABS-2 was downloaded from the GenBank submission on August 23, 2016. A partial sample of the Spiroplasma helicoides as well as a full genomic sequence were downloaded. The steps in analyzing the ORFs of this bacterial species is to extract the long-ORFs from the file, build a training set,or interpolated context model of the genome data (to provide context to Glimmer), and run Glimmer3 to predict the genes in the input sequence.

The commands used to achieve this were as follows:

```
Long-orfs -n -t 1.15 sheli.fasta sheli.longorfs
Extract -t sheli,fasta sheli.longorfs > sheli.train
Build -icm -r sheli.icm < sheli.train
Glimmer3 -o50 -g110 -t30 sheliprt.fasta sheli.icm sheliprt
Extract -t sheliprt.fasta sheliprt.predict > sheliprt.glimmer
```

Detailed information about the different commands used:

The long-orfs command takes in a sequence file and outputs “long, non-overlapping orfs” that can be found within the given sequence. Some of the options selected regarding this command were -n (disregards header in the output) and -t 1.15 (causes output to only display genes with entropy distance scores of < 1.15).

The extract command reads a genome sequence and outputs a fasta file of the regions specified input coordinates.

The build -icm command generates an interpolated context model from the genome data and gives Glimmer context for every position in the reference genome. This gives Glimmer more predictive power as the HMM incorporated within Glimmer is not functioning solely on the previous position in the sequence. The -r option uses the reverse of the input strings to construct the icm model.

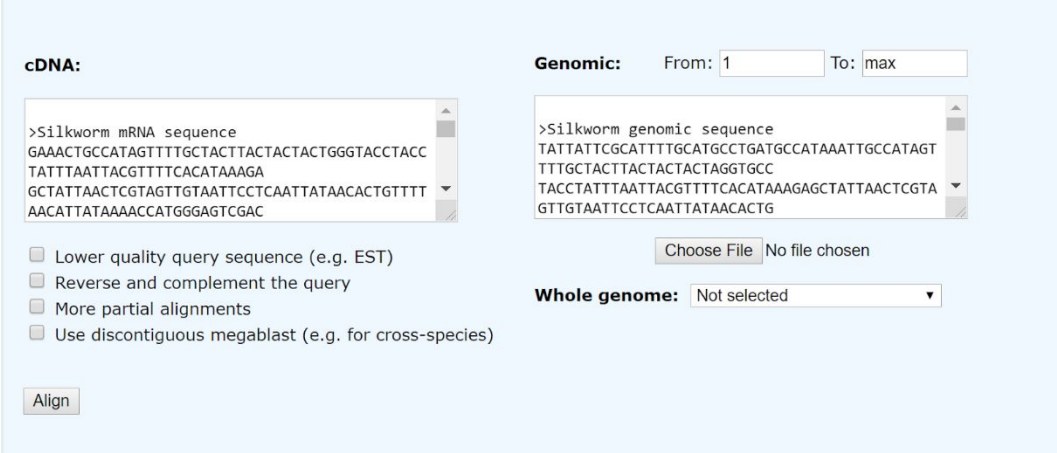
The Glimmer3 command runs the Glimmer predictive function to predict genes in a new input sequence based on the training sequence. The options used were -o50 (sets the max overlap length to 50), -g110 (sets the minimum gene length to 110), and -t30 (sets the threshold score to 30 before it is considered a gene).

Eukaryotic

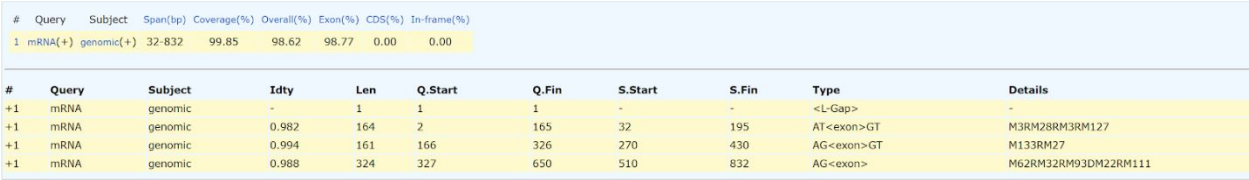
Eukaryotic gene prediction generally falls under expression-based gene predictions that use cDNA and protein sequences aligned to genomic sequences to define the CDS. This method is a bit limited, however as cDNA isn’t always available and that 20-40% of genomes code for rarely expressed DNA. Some examples of expression-based prediction methods are Splign, Genomic BLAST, and BLAT. Splign align cDNA to genomic DNA to find splice junctions and frameshifts as well as alternative gene models. Genomic BLAST allows the user to use ESTs and not need a full cDNA sequence. BLAT is similar to Splign, but the user does not have to specify genomic DNA when comparing. One of the largest issues with gene prediction programs is determining a transcription start site and stop site. De novo gene prediction analyzes genomic DNA to predict exon locations. Some examples of de novo gene prediction programs are GENSCAN, HMMGene, FGENESH, and AUGUSTUS.

Splign Use Example

Using Splign, I aligned a silkworm cDNA sequence and a silkworm genomic sequence to determine their alignment as well as mRNA and CDS coordinates within the genomic DNA.
The first step was to input the raw data into the Splign alignment tool:



The next step was to look at the data visualization and extract mRNA locations and CDS locations from the Splign output. To determine the mRNA locations, we can click on "text" in the upper right hand corner and generate this output:



#	Query	Subject	Span(bp)	Coverage(%)	Overall(%)	Exon(%)	CDS(%)	In-frame(%)
1	mRNA(+)	genomic(+)	32-832	99.85	98.62	98.77	0.00	0.00

#	Query	Subject	Idty	Len	Q.Start	Q.Fin	S.Start	S.Fin	Type	Details
+1	mRNA	genomic	-	1	1	1	-	-	<L-Gap>	-
+1	mRNA	genomic	0.982	164	2	165	32	195	AT<exon>GT	M3RM28RM3RM127
+1	mRNA	genomic	0.994	161	166	326	270	430	AG<exon>GT	M133RM27
+1	mRNA	genomic	0.988	324	327	650	510	832	AG<exon>	M62RM32RM93DM22RM111

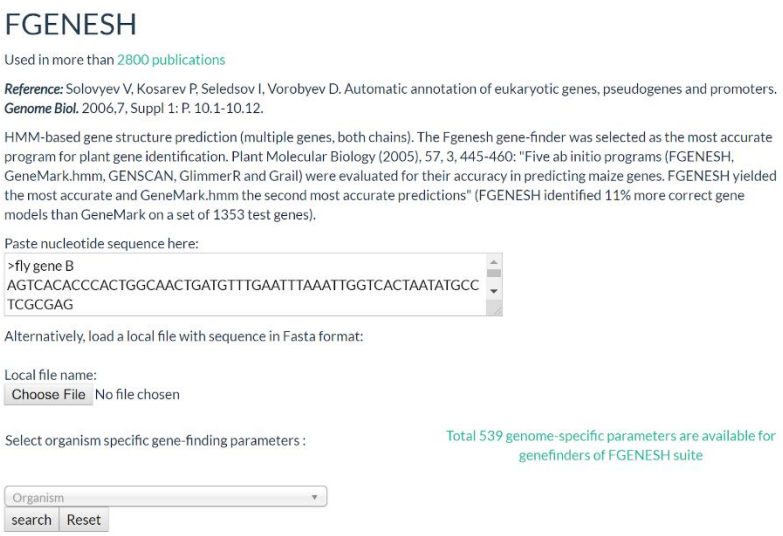
From here, we can disregard the first mRNA as it is only of length 1. The other three mRNA locations are correct. The three mRNA locations in this genomic sequence are from 32~195, 270~430, and 510~832.

To determine the CDS we have to look at the graphical output and count where each segment starts coding for amino acids, as shown by a encoded amino acid in purple above the nucleotides in the photo below:

To determine the CDS we have to count where the coding starts and stops in each separate segment. The CDS for each segment are as follows: 159~195, 270~430, and 510~638.

De Novo Gene Prediction (FGENESH) Use

Using FGENESH, I performed de novo gene prediction. The first step of using FGENESH is to input the nucleotide sequence in question into the software. The sequence I used belongs to a Drosophila species..



After running the FGENESH software, the next step is to analyze the generated output by determining the exon locations as well as the transcription start. The output looked as follows:

```
FGENESH 2.6 Prediction of potential genes in Drosophila genomic DNA
Time    :   Thu May  2 17:28:34 2019
Seq name: fly gene B
Length of sequence: 5544
Number of predicted genes 1: in +chain 1, in -chain 0.
Number of predicted exons 3: in +chain 3, in -chain 0.
Positions of predicted genes and exons: Variant   1 from   1, Score:179.813989
 G Str   Feature   Start        End    Score           ORF          Len

 1 +    1 CDSf       601 -      1429  128.52       601 -      1428    828
 1 +    2 CDSi      4272 -      4522   35.22      4274 -      4522    249
 1 +    3 CDSl      4585 -      4944   35.08      4585 -      4944    360
 1 +      PolA      5023                1.26
```

From this, we can determine that there are 3 predicted exon locations in the given input sequence. CDSf, located from 601~1429, CDSi located from 4272~4522, and CDSl located from 4585~4944. Unfortunately, FGENESH either could not determine a TSS, or there was none present in this particular sequences as there is no specified TSS found in the output.

BED Format

BED format is an incredibly important data format that allows for data to be defined and displayed in annotation tracks using browser-related visualization tools such as UCSC genome browser. BED format stands for Browser Extensible Data and is at its most basic, represented by a 3 column table with each row representing a single entity (ex. exon). The first column is for the chromosome name, the second is for the start position of the entity in question on a chromosome and the third is the end position of the entity. There are many other columns that exist in BED format such as: name of the entity, score, strand, thickStart, thickEnd, etc.

One of the most important things to know about a BED file is whether or not it is a zero-based start or a one-based start. A zero-based start means that the start position of an entity is counted as starting at 0, and the end position of that entity is not counted. A one-based start starts counting at the start position of an entity at 1 and includes the end position of that entity in the count.

To create a BED file from scratch is simple, yet very necessary to know. To start, open a text browser and create at least three columns and save it with a .bed extension. Each column is separated by tabs. Each column MUST have the preceding column before it filled in with data. For example, if I wanted to have the "strand" column in my bed file, I would need to not only include the original three BED columns, but also the name and score columns.

BED Format Use Example

Using BED format, I created a BED6 file with two lines/entities based on a paper detailing the ddlR-ddl operon found in Brevibacillus brevis. The given ddlR CDS location is 2968133..2969623. Given this information, I wanted to find the promoter region and the 5' UTR region. The promoter region is 170bp in length and starts 140 nucleotides upstream from the start codon and ends 29 nucleotides downstream of the start codon. The predicted 5' UTR begins 38 nucleotides upstream from the start codon.

The first step in solving this problem is to understand zero-based start as being inclusive in start of the counting and exclusive in the end of the counting. TO generate a zero-based BED6 file and to determine the start location and end location of the promoter, I counted 140 nucleotides upstream of the start codon at 2968133, giving me 2967992 and then I counted 29 nucleotides downstream of the start codon, giving me 2968162. Thus, the promoter region in bed format is:

```
Chr1  2967992    2968162    Promoter   0     +
```

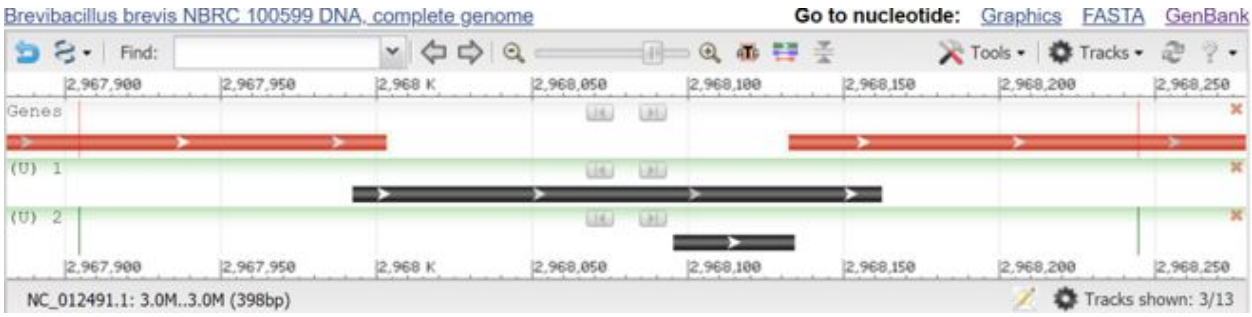
The 5'UTR was found in a similar fashion by counting 38 nucleotides upstream from the start codon and ending it at the nucleotide that immediately precedes the start codon. The 5'UTR was identified as follows:

```
Chr1  2968085    2968134    5'_UTR     0     +
```

The combined BED file reads:

```
Chr1  2967992    2968162    Promoter   0     +
Chr1  2968085    2968134    5'_UTR     0     +
```

After generation of the BED file, I then uploaded it into the NCBI Genome Browser for Brevibacillus brevis to visualize the data in the context of the Brevibacillus brevis genome. The following is after I had uploaded the BED file to the NCBI genome browser:



Other Important Data Formats

Wig Format

Wig Format is generally used to display continuous-valued data. It is a format that can display continuous data such as GC percent, probability scores, or transcriptome data. Wig format can have two different sub-formatting options: variableStep or fixedStep. VariableStep is Wig format designed for irregular intervals between each data point. FixedStep is for data with regular intervals between each data point and is much more compact that variableStep.

SAM Format

SAM format stands for Sequence Alignment/Map format. It is tab-delimited, much like BED format. It is comprised of a header section and an alignment section. Header sections being with "@" where alignment sections do not. SAM is a text format to store sequence data in ASCII. Each alignment section has: a query name, bitwise set of information to describe the alignment such as multiple fragments, unmapped, reverse strand, etc. It can also contain more information such as the reference sequence name, the left-most position of the alignment, mapping quality, etc.

BAM Format

BAM format is the compressed version of SAM format. It takes the SAM format of index-able nucleotide sequence alignments and converts them to binary. The big advantage BAM has over SAM is that only specific portions of the BAM file are needed to generate a display on visualization tools such as UCSC. It can thus, display very large files easily where in any other format it would probably time out trying to upload them.

FASTQ Format

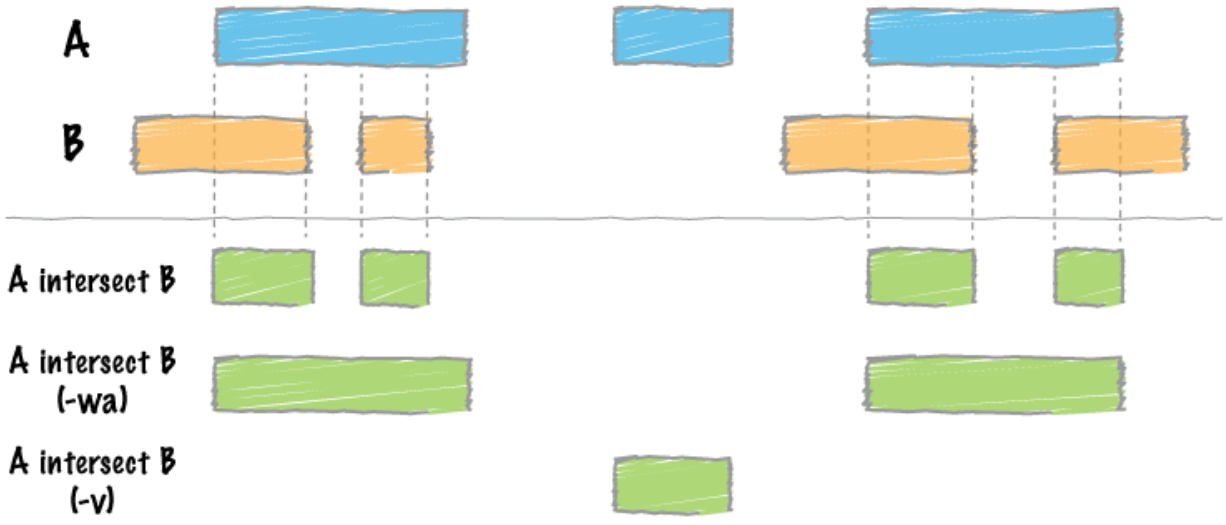
FASTQ Format is a text file that displays Next Generation Sequencing information. It shows nucleotides that are incorporated at each position as well as displays a phred score, which shows how confident we are that the nucleotide at that position was sequenced correctly. FASTQ Format is comprised of an "@" header, a DNA sequence, a "+" header, and a phred quality score.

BED Tools

BED tools can be used either via the command line if BEDtools is installed or can be run using Galaxy. BEDtool functions include: intersect, cluster, subtract, and sort (amongst many other functions). Intersect allows to find overlapping intervals in various ways using overlap. Cluster/sort can group and sort intervals, but does not merge overlapping/nearby intervals. Subtract has two options of "no overlap" and "non-overlapping". No overlap shows intervals that do not have any section of the first dataset overlapping with any section of the second dataset while non-overlapping shows the intervals of the first dataset with the portions of the second dataset that overlap removed.

BEDtools Use Example

Using the BEDtools available on Galaxy, I took two bed files containing exon information from chromosome 5 q31 and CpG regions in chromosome 5 q31. I then used various applications of BEDtools to determine unique exons that intersect and do not intersect a CpG region, as well as unique CpG regions that intersect and do not intersect an exon. To accomplish this, I first loaded my files onto Galaxy using the upload file protocol. I then selected BEDtools to start my analysis. To first determine the unique exons that intersect with a CpG region, I selected the non-CpG dataset as entry A with the CpG dataset as entry B. I then selected the option to write entry A once if there were overlaps found in B. To determine the unique exons that do not intersect with a CpG regions, I left that option empty and allowed the program to display those results that do not overlap at all with B. To determine whether or not an unique CpG region intersected or did not intersect with an exon, I simply reversed which was dataset A and which was dataset B. A useful visual representation of the processes involved in BEDtools intersect can be seen below, from the Galaxy website.



SAM tools

SAMtools are used either on the command line or through Galaxy to convert SAM files into BAM files. The main uses of SAMtools are to convert SAM files to BAM files, sort BAM files, and index sorted BAM files, thus creating an indexed BAM file. The three commands for SAMtools are as follows:

```
Samtools view - bS filename.sam > filename.bam
```

This converts a specified SAM file to a BAM file. Can also be done use Galaxy, which does not involve any "commands".

```
Samtools sort filename.bam filename.sorted
```

This command sorts a BAM file based off of chromosomal position. The output file is shown as a "sorted" file.

```
Samtools index filename.sorted.bam
```

This command indexes a sorted BAM file to create a BAM index file for use.

R Tools

R is a free software environment for the development of statistical analysis tools and packages that can be shared openly with other users. In the case of bioinformatics, R is an extremely useful tool for the analysis of large data sets (such as genomes) using a variety of different packages that have been developed specifically for bioinformatics application. The example that I will be showing in this section is biomaRt. BiomaRt is part of the bioconductor package and it works by accessing the Ensembl database (in addition to some other databases) to retrieve information specified by a user's query. Much like the online version of Biomart found on the Ensembl website, biomaRt can filter data using many categories, search up various attributes, and narrow down queries using specific values. BiomaRt's strength comes in its ability to access multiple databases and extract massive amounts of information depending on the user's queries.

R Tools: biomaRt Use Example

Using biomaRt, I wished to retrieve two tables of information regarding Huntington's disease. The first table of information I wished to gather was in regards to 5 MIM values that I had found by searching OMIM for Huntington's disease. These 5 MIM values were the first five entries that had to do with Huntington's disease according to OMIM. I wished to retrieve information regarding the attributes: Entrez Gene ID, HGNC symbol, and the Ensembl Gene ID. The query that I structured to retrieve the prior mentioned data was as follows:

```
> getBM(attributes=c("entrezgene", "hgnc_symbol", "ensembl_gene_id"),
filters=c('mim_morbid_accession', 'with_mim_morbid'), values =
list(c("143100", "613004","604802","606438","603218"),TRUE),
mart=ensembl)
```

The getBM command pulls data from a "mart" or an online database that biomaRt has access to. From there, attributes are selected according to their specific attribute "name". After the attributes have been named (the information our query will compile into an output table), we can then use filters to narrow the output. Each filter needs to have values attached to them to cut down on the superfluous information that we may not be interested in. Lastly, we select our "mart" to be ensembl. This getBM command returns the following table of information:

	entrezgene	hgnc_symbol	ensembl_gene_id
1	3064	HTT	ENSG00000197386
2	5621	PRNP	ENSG00000171867
3	57338	JPH3	ENSG00000154118

From the information retrieved by the getBM command, I was able to determine the Entrez gene ID, HGNC symbol, and Ensembl gene ID while limiting my query to only those results that had the MIM accession numbers I had found previously from OMIM.

The second getBM command that I ran was to find all of the Ensembl Transcript IDs in addition to the HGNC symbol and Ensembl gene IDs while limiting my search to the same OMIM accession numbers.. The command was as follows:

```
> getBM(attributes=c("hgnc_symbol",
"ensembl_gene_id","ensembl_transcript_id"),
filters=c('mim_morbid_accession', 'with_mim_morbid'), values =
list(c("143100", "613004","604802","606438","603218"),TRUE),
mart=ensembl)
```

This query resulted in the following table:

	hgnc_symbol	ensembl_gene_id	ensembl_transcript_id
1	HTT	ENSG00000197386	ENST00000647962
2	HTT	ENSG00000197386	ENST00000649900
3	HTT	ENSG00000197386	ENST00000355072
4	HTT	ENSG00000197386	ENST00000648150
5	HTT	ENSG00000197386	ENST00000506137
6	HTT	ENSG00000197386	ENST00000512909
7	HTT	ENSG00000197386	ENST00000510626
8	HTT	ENSG00000197386	ENST00000649009
9	HTT	ENSG00000197386	ENST00000649131
10	HTT	ENSG00000197386	ENST00000509618
11	HTT	ENSG00000197386	ENST00000650595
12	HTT	ENSG00000197386	ENST00000650588
13	HTT	ENSG00000197386	ENST00000513639
14	HTT	ENSG00000197386	ENST00000513326
15	HTT	ENSG00000197386	ENST00000509043
16	HTT	ENSG00000197386	ENST00000502820
17	HTT	ENSG00000197386	ENST00000509751
18	HTT	ENSG00000197386	ENST00000512068
19	HTT	ENSG00000197386	ENST00000513806
20	HTT	ENSG00000197386	ENST00000508321
21	PRNP	ENSG00000171867	ENST00000379440
22	PRNP	ENSG00000171867	ENST00000430350
23	PRNP	ENSG00000171867	ENST00000424424
24	PRNP	ENSG00000171867	ENST00000457586
25	JPH3	ENSG00000154118	ENST00000537256
26	JPH3	ENSG00000154118	ENST00000301008
27	JPH3	ENSG00000154118	ENST00000284262
28	JPH3	ENSG00000154118	ENST00000563609

This table shows all of the different Ensembl transcript IDs for the given OMIM accession numbers related to Huntington's disease. BiomaRt is an extremely effective tool in searching through various bioinformatics databases and can be used to retrieve relevant information.

ncRNA

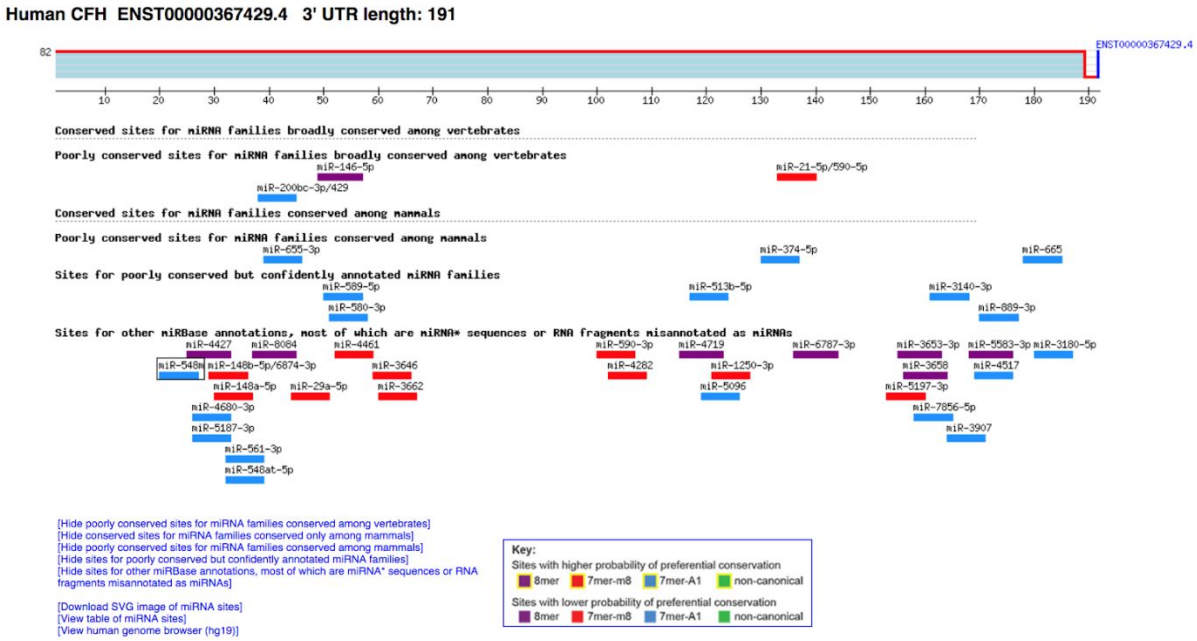
ncRNA is any RNA that is not translated to protein. This includes: transfer RNA, ribosomal RNA, Long noncoding RNA, small nuclear RNA, microRNA, short interfering RNA, and guide RNA. Functional RNA is similar to ncRNA, but has a broader connotation. Also includes secondary structures found in mRNA such as riboswitches and SECIS elements. There are a variety of tools and databases that can help with the prediction of ncRNA elements, mainly miRNA. I have included two tables detailing each tool's or database's particular purpose.

Tool	**Purpose**
PPfold	Multiple alignment comparisons of RNA
RNAz	Predicts fRNA based off of MNE (minimum free energy). MFEs of a region calculated by shuffling a sequence repeated to determine which combination has the lowest free energy
PicTar	Algorithm to find miRNA targets
DIANA-TarBase 7.0	Prediction algorithm for miRNA-mRNA targeting

Database	**Information Provided**
TargetScan	Search species and genes for calculations on miRNA targets
miRNEST 2.0	Plant and animal miRNA with data from deep sequencing
PolymiRTS	Looks at SNPs in miRNA target sites
miRDB	Search for miRNAs that might target a given mRNA or search for mRNA targets given a miRNA
miRBase	Database of known miRNA sequences

miRNA Prediction Use Example: TargetScan

Using TargetScan, I searched to find the predicted miRNA binding sites for the CFH gene. The CFH gene encodes for the Complement Factor H protein, which is a key component in the complement pathway (immune response). From my search of CFH, the following output was generated:



This image displays: conserved sites for miRNA families amongst vertebrates, poorly conserved sites amongst vertebrates, conserved miRNA families in mammals, poorly conserved families in mammals, sites for poorly conserved annotated miRNA families and sites for other miRBase annotations. TargetScan is a great tool to find conserved and poorly conserved sites for miRNA families amongst different organisms and can help predict where certain miRNAs might interact with specific genes of interest.

Galaxy

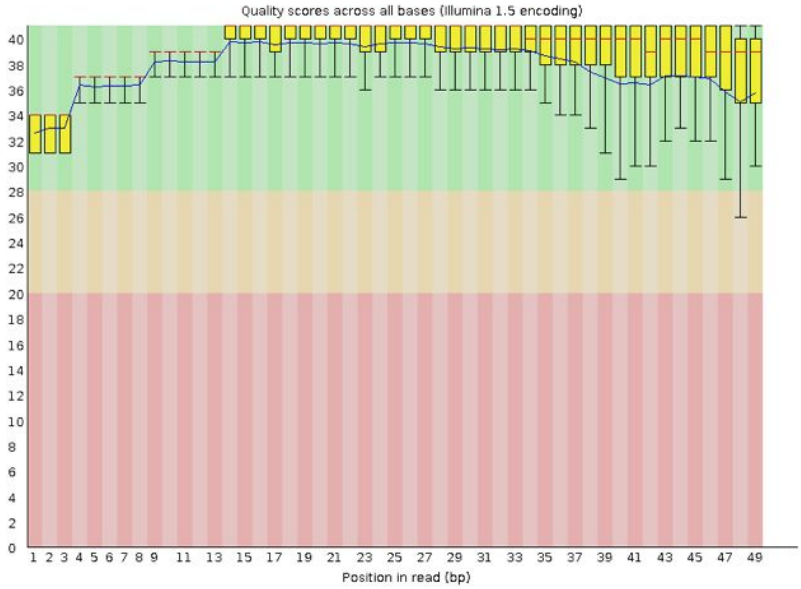
Galaxy was designed to be a source for both trained and untrained bioinformaticians or scientists for comparative genomics. The major goals of Galaxy are to identify functional genes, determine biological roles of these genes, and to understand their evolutionary history. Galaxy is an excellent tool to use for large selection of bioinformatics processes as it has softwares integrated in an online format that is easy to use and understand. Galaxy also hosts its own servers to run processing, so even users without powerful computing or data storage have access to these processes. It can also convert data from one format to another as well as export data to other visualization tools. A great aspect of Galaxy is the creation of workflows, meaning that once you design a particular workflow, you can run data through it without having to manually set the parameters every time. It also allows others to visualize the steps taken to achieve certain data analysis.

A few of the more important processes using Galaxy that I had a chance to use were Next Generation Sequencing, ChIP-seq analysis following NGS analysis, and RNA-seq analysis. NGS relies on DNA synthesis from a template to detect the nucleotide that is next incorporated at each position. A phred score shows how confident we are that the nucleotide that was sequenced is correct. The sample data is then aligned to a reference genome using BWA, Bowtie2, or HISAT, and an output BAM file is created where variants can be identified using FreeBayes. Once variants have been identified, the VCFfilter tool can be used to filter the variants using different attributes.

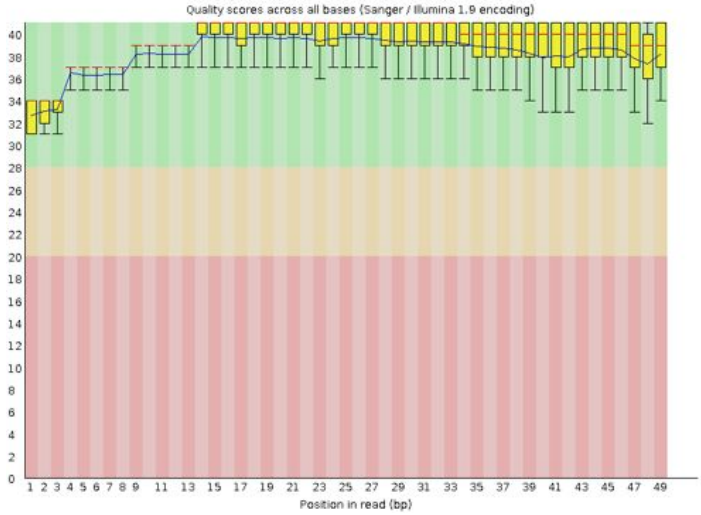
ChIP-seq is based on using the ChIP method to precipitate proteins using antibodies that specifically bind to those proteins. It not only precipitates the histone protein, but also the DNA that is bound to it. ChIP-seq is when ChIP is followed by NGS. Antibodies precipitate the protein-DNA complexes and the DNA that is bound to these histones are then sequenced and aligned to a reference genome. ChIP-seq data is generally found in FASTQ format and can be aligned using Bowtie2 or BWA. Data from ChIP-seq comes from peak calling, where peaks are identified in aligned datasets to show DNA regions that had a target protein that was immunoprecipitated. RNA-seq is massive parallel cDNA sequencing. Analyzing RNA-seq data can lead to defining a TSS, finding sncRNA and lncRNAs, finding alternative splice sites, and quantifying transcript levels. It can be both referenced-based and de novo. Reference-based aligns RNA-seq data to a reference genome, but this can have issues as cDNA mainly comes from spliced mRNA. As such, when it is aligned with genomic DNA, large gaps can occur which show intronic regions. This leads to the necessity of specialized RNA-seq alignment tools such as TopHat.

NGS Data Preparation Use Example

Using NGS analysis tools found on Galaxy, I ran a sample NGS data file from an experiment on C. elegans. To start, I first uploaded the sample file to Galaxy through their upload data protocol. In doing so, I also selected the latest C. elegans as the species genome. The second step was to run a **FASTQC on the sample file to determine the quality scores** of the sample data. Included below is an image for the FASTQC boxplot of quality scores:



Looking at the boxplot data, all of the sequences (for the most part) have their entirety in the green quality scores, meaning that this data is all of good quality. The encoding scheme that was used in this FASTQC was of Illumina 1.5 with read lengths of 49 over 20000. To get this data into the Sanger/Illumina 1.9 encoding scheme, I reran the data through the **FASTQ Groomer** to change the encoding scheme from Illumina 1.5 to Sanger/Illumina 1.9. My next step was to trim the data using a sliding window of 4 bases and trim the data until each window had an average quality score of > 30 using the **FASTQ Quality Trimmer** protocol. I then reran FASTQC to see how my data had changed after grooming and trimming:



The quality of the sequences in my sample after grooming and trimming had an upwards trend in phred scores compared to the non-groomed/trimmed phred scores. This meant that after my grooming and trimming processes, my data had a better "quality" of sequences ready to be analyzed.

NGS Analysis Use Example

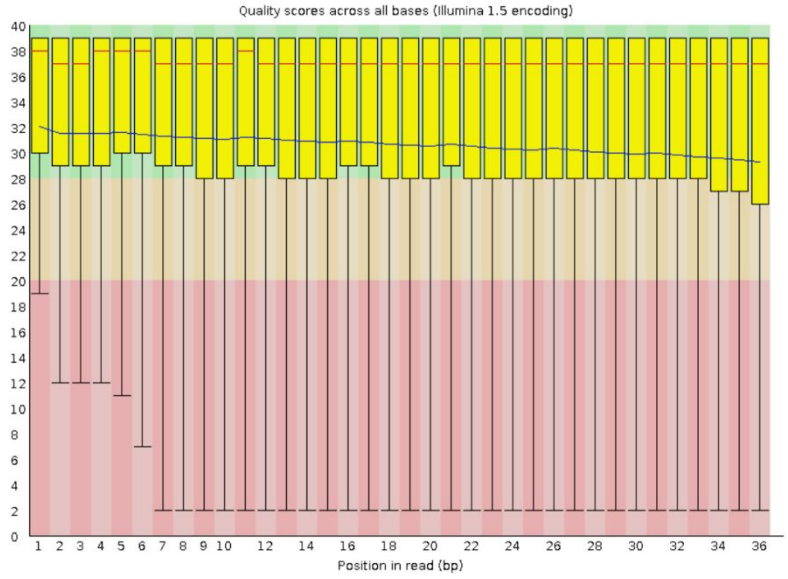
Using Galaxy tools, I identified SNPs in sample NGS data from the 1000 Genomes Project using two FASTQ files: a forward read and a reverse read to represent a paired-end sequence. I prepared the data similar to the example above: I ran a FASTQC on the files and found them to have the

Sanger/Illumina 1.9 encoding (so there was no need to run FASTQ Groomer) and I then ran a FASTQ Quality Trimmer to remove low quality bases with window size of 4 and average quality per window of > 20. I then ran an **alignment to a reference genome using BWA** setting the reference genome to hg19. Once I had the alignment to a reference genome, I then ran the **FreeBayes tool to identify variants** found in chr22:0-51304566. After running FreeBayes, the **output VCF file showed that there are 53 variants** found in that particular region of chromosome 22, along with all of them having a corresponding RefSeq gene. To got a little further in-depth, I chose an SNP at random, located at chr22:18881837 and took a closer look. This SNP changes a C nucleotide to a T nucleotide in the LINC01663 intron 1/11. The LINC01663 gene is a long intergenic non-coding protein RNA, meaning that the SNP I selected does not have an effect on gene function, as the gene only produced a non-coding RNA. However, this analysis technique can be applied to other SNPs of interest to help determine changes in gene function due to the variants discovered via NGS analysis.

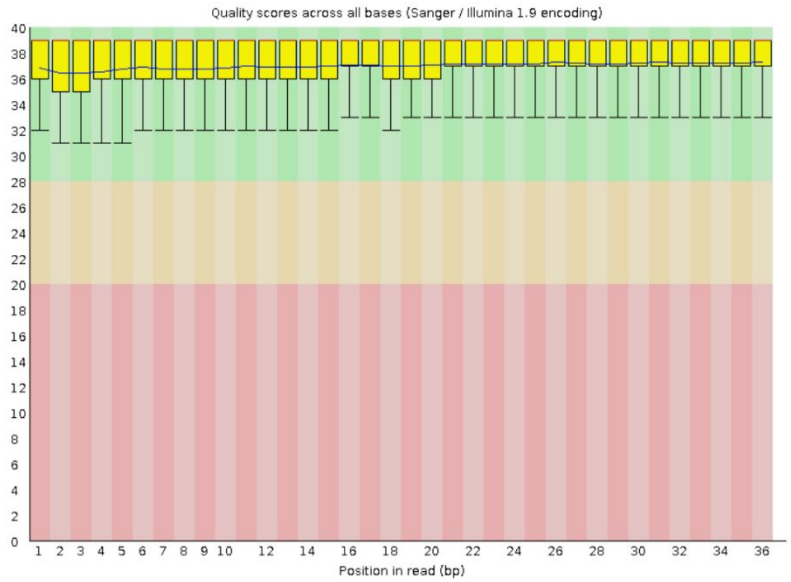
ChIP-seq Analysis Use Example

Using Galaxy tools, I performed ChIP-seq analysis to determine peaks, showing where in aligned DNA regions a target protein was immunoprecipitated. The process of ChIP-seq analysis is very similar to that of the normal NGS analysis to prepare the sequence data for analysis. My example of ChIP-seq analysis took a FASTQ file of ChIP-seq reads from Mus musculus (mm9) and downsampled it to a subset of reads from chr19. The first step to prepare ChIP-seq data was to run it through FASTQC to determine the quality score encoding, and run it through the FASTQ Groomer to convert it to Sanger/Illumina 1.9. I then ran Trimmomatic on the groomed data using sliding windows set to 4 and minimum average quality score set to 25. I then reran my data through FASTQC to see the differences between the quality scores and the encoding scheme.

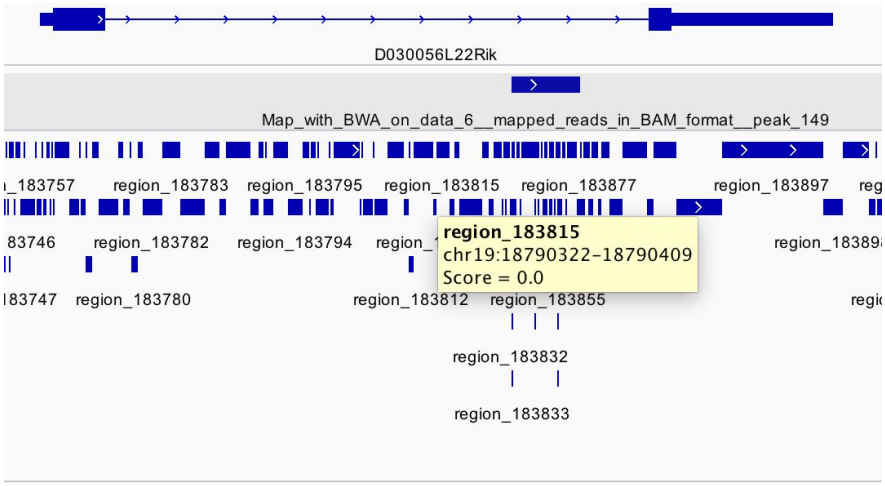
Before trim:



After trim:



As can be seen, after running Trimmomatic, the sequence data quality drastically increased compared to before the trim. I next ran my trimmed data through the mapping/alignment tool BWA, using the reference genome of mm9. Once I had my sequences aligned to the reference genome, I then ran the **MACS2 callpeak protocol** on my BWA output BAM file. I then loaded the MACS2 Bedgraph treatment file and narrow peaks BED file (both from the MACS2 callpeak protocol) to IGV and aligned them to the mm9 reference genome. Below is an image of a location where a gene is displayed with both narrow peaks, which displays the peak summit, and a bedgraph of treatment pileups, which displays the peak coverage.

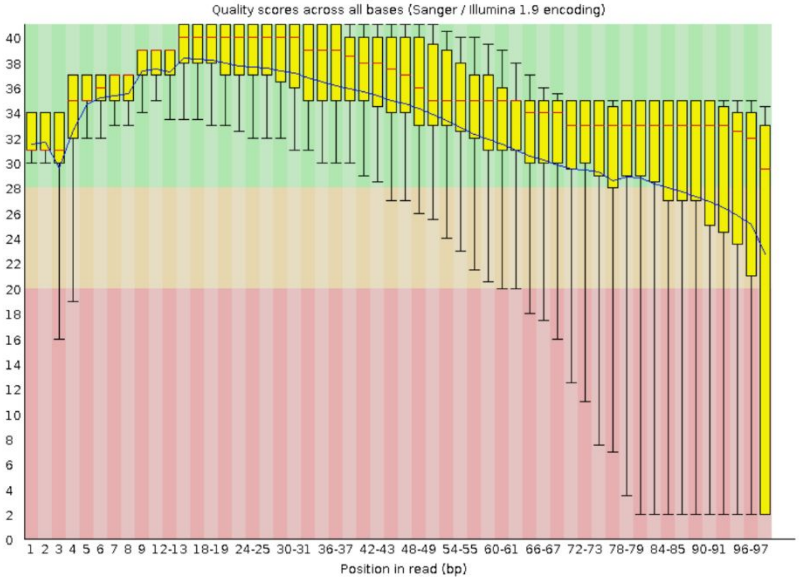


Using this information, I am able to determine where in the mouse genome our ChIP-seq data shows bound DNA and can extrapolate from there depending on what particular protein was bound. For example, if this the protein in question from this ChIP-seq data was in regards to histone modifications, we would be able to determine that in the areas of the peaks, there are histone modifications to the bound DNA in that particular location.

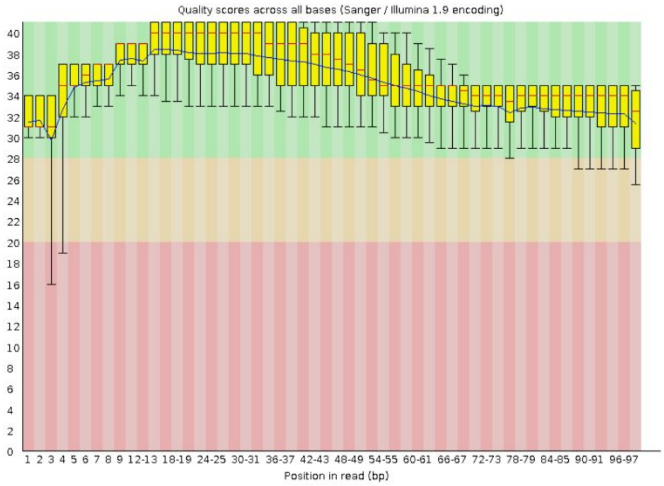
RNA-seq Analysis Use Example

Using Galaxy tools, I performed RNA-seq analysis on a set of two different types of cellular states, G1E and megakaryocyte to identify differentially expressed transcripts between the two states. The

first step in this process was uploading the data onto Galaxy. Each data set had 2 replicates (meaning 2 replicates for G1E and 2 for megakaryocyte) and I ran a FASTQC protocol on them to determine their read quality and then Trimmomatic (an analog to FASTQ Quality Trimmer) to remove the low quality bases from these sequences. I then ran another FASTQC on the trimmed reads to see the differences. Before trim example of G1E rep #1:



After trim example of G1E rep #1:

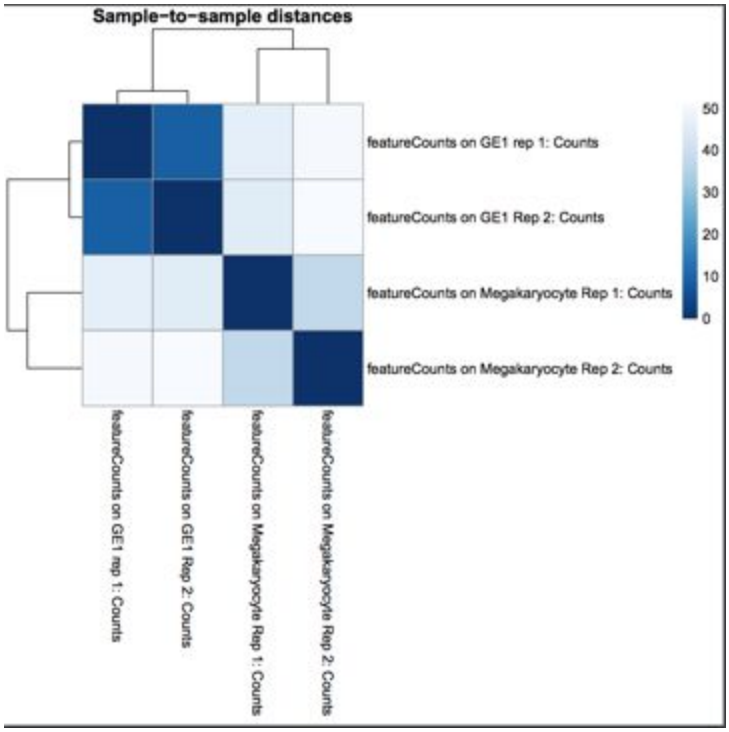


After trimming my samples and rerunning **FASTQC**, I then went on to the next step in my RNA-seq analysis, which is that of aligning my sequences (G1E and megakaryocyte) to the Mus musculus (mm10) mouse genome. To do this, I used the **HISAT2 alignment program** to align my trimmed G1E and megakaryocyte sequences while setting some of the advanced options. The options selected were: penalty for non-canonical splice sites set to "3", penalty function for long introns with canonical splice sites set to "0.0", and transcriptome assembly reporting set to report alignments tailored for transcript assemblers including **StringTie**. From this, I had four mapped reads (two for G1E and two for megakaryocyte) that I then wanted to perform **de novo transcript reconstruction** with. In doing this, I r**econstructed a complete transcriptome** identification from my experimental samples. To achieve de novo transcript reconstruction, I ran all four aligned samples through Stringtie. **Stringtie generates four transcriptomes that represent the RNA-seq samples that I have been analyzing**. These transcriptomes can be merged

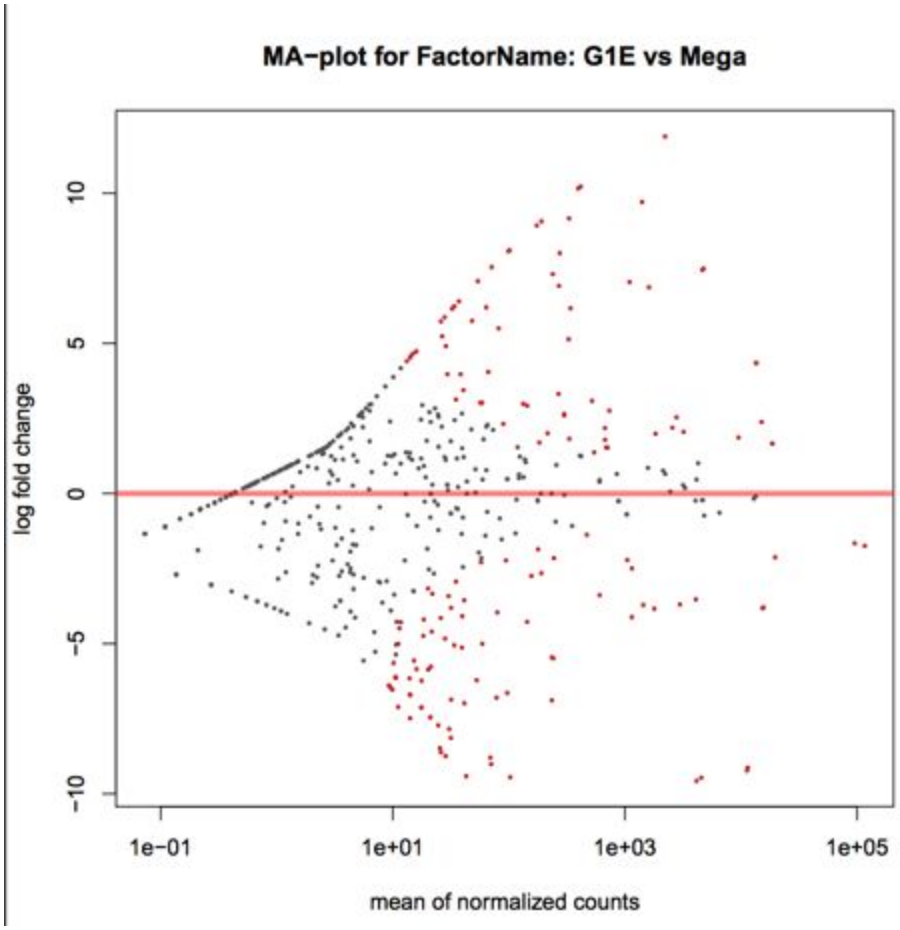
together, using Stringtie - Merge to remove redundant transcripts across the samples and the RefSeq reference. I then used **GFFCompare** to make annotations of the transcripts in my newly generated transcriptome for their relations to every transcript found in the RefSeq reference.

Now that my transcriptome database is complete, the real analysis of differential expression between the two cell states can occur. The first step in analyzing our data is to count the number of features in each of my HISAT2 aligned reads compared to our GFFCompare transcriptome database using **FeatureCounts**. **The feature (or read) counts estimate the a particular transcript's expression.** The second step in our data analysis is the run **DESeq2**, **a tool used for differential gene expression**. DESeq2 takes in data from FeatureCounts as read counts and outputs: gene identifiers, mean normalized counts, log-base 2 fold change, standard error of log2 fold change, Wald statistic, p-value for statistical significance of a change, and p-value adjusted for testing with Bejamini-Hochberg procedure. It also outputs a graphical summary which is a bit easier to understand. Some of the example graphical output is below:

This graph shows that the replicates between each set of cell states agree to a decent amount, however between the two sets, the RNA-seq data does not agree. Distances between GE1 and megakaryocyte is large compared to the distance between the replicates.



This graph shows the log-fold change between the factors analyzed G1E and Megakaryocyte differ pretty greatly from one another.



In addition to graphical analysis, running the **Filter tool on the DESEQ2 output file allowed me to determine certain factors about our sample sequences**, such as the amount of up-regulated and down-regulated transcripts that can be found in G1E with a p-value of < 0.01. When running this filter, I found that there are 77 transcripts with significant change in expression between G1E and megakaryocyte with 39 upregulated and 38 downregulated. Additionally, I chose a random transcript with a log2 fold change of at least 2 to learn more about it. The transcript that I chose was NM_008272 and it encodes for the Hoxc9 mRNA transcript. The Hoxc9 gene belongs to the homeobox family of genes which encodes for transcription factors used in DNA-binding and morphogenesis. It is highly expressed in mammary glands and ovary tissue in adult mice. From this example of RNA-seq analysis, I was able to determine important differential expression patterns between two cell states from Mus musculus. Doing this sort of analysis from other samples and experiments could prove extremely beneficial in differential expression research.

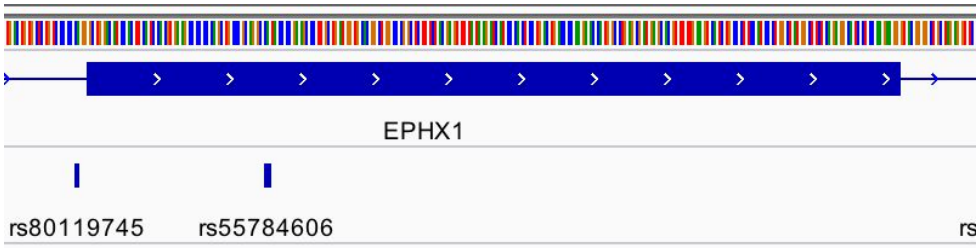
IGV - Visualization

IGV stands for Integrative Genomics Viewer. It is used to handle large genomic-sized data and can be used locally to allow for bigWig and bigBed formats. The goal of IGV is to make data visualization for trained and untrained users easy and intuitive. IGV can upload both BED and WIG files to be visualized alongside stored genomic data such as hg38, hg19, mm9, etc.

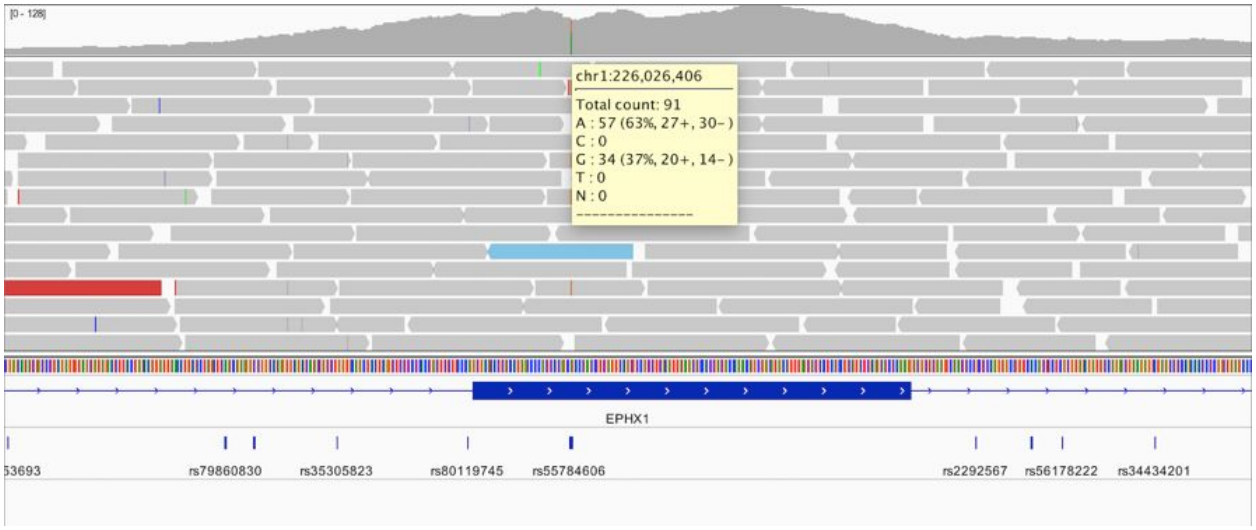
IGV Use Example: SNPs and CNVs

Using IGV, I wanted to find out information regarding the gene EPHX1 and the presence of SNPs within this gene. To accomplish this, I loaded the hg19 human genome as well as dbSNP 1.3.1 (from 1000 Genomes Project) onto IGV and selected EPHX1 to look at. Once I had loaded the correct genome and dbSNP file and zoomed in on EPHX1, this is what I saw:

To find out more about the SNPs present in this gene, I decided to only look at exon 4. A zoomed in image of exon 4 and the SNPs present (according to dbSNP 1.3.1) is as follows:

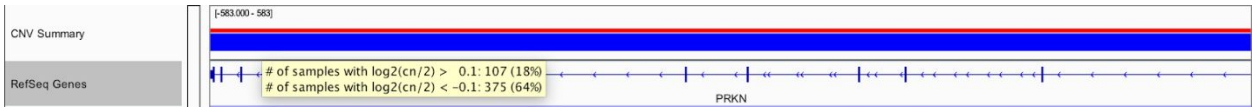


There is only a single SNP that directly overlaps with exon 4 of the EPHX1 gene and it has the ID rs55784606. I then wanted to see the count of nucleotides at this particular position for all of the genomes sequenced for the hg19 genome track. To accomplish this, I loaded the HG00096 exome coverage track and this is what I saw:



From this image, I was able to discern that there were a total of 91 counts of nucleotides at the position of rs55784606 with 57 of them being A and 34 of them being G.

The next use case of IGV was to compare ovarian cancer and breast cancer in a select few genes to evaluate the CNV summary between the two in each gene region. The genes of interest were PKN2, GRXCR1, PRKN, and PPIAL4A. To examine the differences in CNV summaries between these two cancers, I first loaded data from OV-TP (ovarian cancer) to IGV and then searched the genes in question. Here is an example output for the PRKN gene and its associated CNVs:

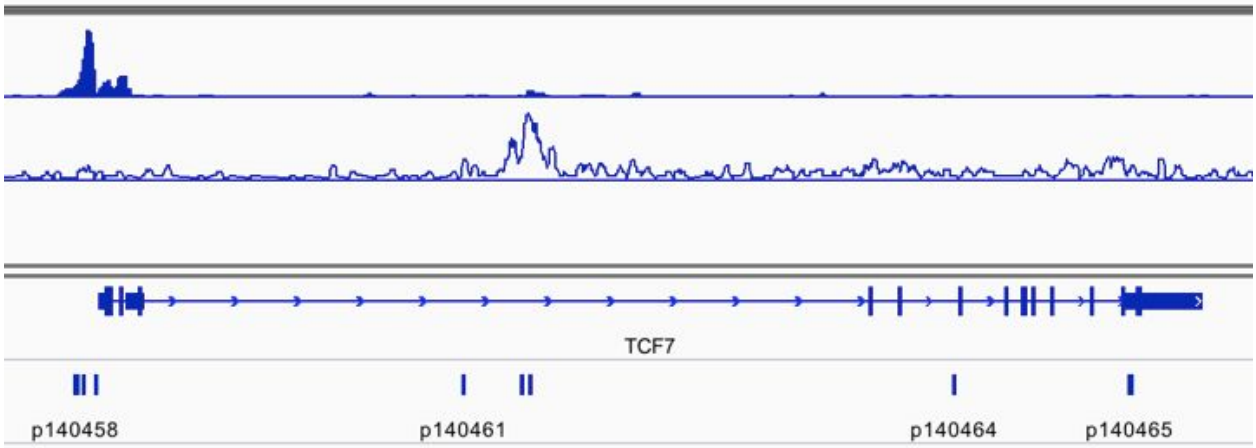


As can be seen, the PRKN gene has significantly more blue than red in its CNV summary. The colors represent either a gain in function (red) or a loss of function (blue) for patients with that CNV. To sum up my results, I created a table for easy reference on the differences found between ovarian cancer and breast cancer in four genes:

Gene	OV-TP	BRCA-TP	Summary
PKN2	More than twice as much red as blue	Twice as much blue as red	OV and BRCA are more or less opposites when it comes to CNVs in this gene.
GRXCR1	Slightly more blue than red	Twice as much blue as red	Both trend towards more blue in this gene, but BRCA seems to have more blue than in OV.
PRKN	Mostly blue	*IGV did NOT work when searching this gene with BRCA-TP as the data set*	*IGV did NOT work when searching this gene with BRCA-TP as the data set*
PPIAL4A	Overwhelmingly red	Overwhelmingly red	Both OV and BRCA seem to be similar in trending towards red.

IGV Use Example: ENCODE Data

Another use case for IGV is to visualize data found in ENCODE, the Encyclopedia of DNA elements. The data of interest in this case was DNase hypersensitivity (DSH) and layered H3K4me3 in the HUVEC cell line. I first used the UCSC Table Browser to retrieve the data from these ENCODE tracks from the region of chr5:134000000-134250000 and loaded these two tracks to onto IGV using hg38 as my reference genome. I then chose to look at the TCF7 gene and visualize the DSH and H3K4me3 data.



As can be seen in this image, the top graph is that of layered H3K4me3 and the graph below TCF7 is that of the DHS regions. There are 8 DSH regions that are located close to the TCF7 gene. P140458 and P140459 are both upstream of the TSS, P140460 is associated with exon 1, P140461, P140462, and P140463 are all located in an intronic region, P140464 is located right before exon 6, and P140465 is located on exon 9. These regions show areas of DNase hypersensitivity, thus showing areas in which chromatin is no longer as condense and is accessible for transcription. The layered H3K4me3 displays quite a bit of H3K4me3, a histone modification that is most often associated with the TSS of actively transcribed genes. From this information, I garnered that TCF7 is most likely actively transcribed in the HUVEC cell line as the TSS displays histone modifications that are found in actively transcribed genes and there are many areas of DSH throughout the gene, leading me to believe that the chromatin surrounding this gene is open and less condensed.

References

1. Yun, S. and Yun, S. Bioinformatics: Tools for Genome Analysis AS.410.635.81. Johns Hopkins University. Maryland. Spring 2019.
2. "The UCSC Genome Browser Coordinate Counting Systems." *UCSC Genome Browser Blog*, 19 Dec. 2016, genome.ucsc.edu/blog/the-ucsc-genome-browser-coordinate-counting-systems/.
3. "Frequently Asked Questions: Data File Formats." *Genome Browser FAQ*, genome.ucsc.edu/FAQ/FAQformat.html.
4. "Galaxy." *Galaxy*, usegalaxy.org/.